

1 FORMAL PROOF METHODS FOR ANALYZING CIRCUIT

2 LOADING PROBLEMS UNDER OPERATING CONDITIONS

3 Field of the Invention

4 This invention relates to the art of logic design and implementation
5 and, more particularly, to a tool for determining if any driving node in a very
6 complex logic circuit, such as a processor, has suitable power handling
7 capacity to drive its collective loads under “real”, rather than “worst-case”
8 assumed, conditions.

9 Background of the Invention

10 Modern integrated circuits have become very dense and complex. For
11 example, state-of-the art integrated circuits constituting processors may each
12 include many millions of transistors and other components. There are
13 software logic equation generation and analysis tools for extracting logic
14 equations from a circuit description and providing certain preliminary
15 analysis. This is a routine procedure carried out early in the logic design of a
16 complex digital system component. However, checking the resulting logic
17 equations of, for example, a processor for subtle errors is a very formidable
18 problem which is impossible to carry out manually and difficult to achieve
19 with any test software.

1 It might be thought that all possible errors could be found by
2 exercising exhaustive combinations of inputs, outputs and operations
3 sequentially, with test software, on the logic equations representing the
4 design, but, even with a high speed test program running on a powerful
5 machine, the time required is prohibitive and adequate coverage cannot be
6 achieved. As a result, a class of advanced logic analysis programs have been
7 developed which first examine the logic equations and reduce them before
8 analyzing all possible combinations in a search for possible design errors.
9 The processes of these advanced analysis programs are variously known as
10 formal proof, equivalence proof, equivalence checking, formal verification,
11 formal validation, property checking, etc. (For convenience, the term
12 “formal proof” is used herein as a generic term for these types of programs.)
13 While formal proof programs may still take a long time to analyze all
14 combinations of the reduced logic, the time is not prohibitive for use during
15 the development and analysis of a complex logic component.

16 In the preliminary analysis phase of logic circuits by logic equation
17 generation and analysis programs, violations of “loading” and “drive”
18 restrictions for driving nodes and other similar analyses are made. These
19 analyses are typically based upon absolute worst-case assumptions which is
20 necessary because the program doing the analysis typically does not “know”

1 the logic states or have a list of the legal conditions which might control or
2 restrict the operation of the logic circuit under “real” conditions. These
3 absolute “worst case” assumptions involve no analysis of the logical state of
4 a circuit and simply look for violations of loading rules while making the
5 assumption that all possible loads are “actively loading” a specific driving
6 circuit at all times. But, this absolute worst-case assumed solution may be
7 much too pessimistic in that, in all “real” possible combinations for a given
8 driving node, some of the loads may be through transistors which are not
9 enabled such that the actual worst case loading might be less or even much
10 less. This characteristic leads to warnings or error reports, from circuit
11 analysis programs, which are in fact not “real” or possible during the actual
12 operation of the logic circuit. The logic even of a single driving node may be
13 too complicated to analyze easily by hand, and/or the number of
14 warnings/errors reported may obscure the recognition of “real” errors; i.e.,
15 the important message may be buried in a report containing a large number of
16 “false” errors.

17 While the absolute worst-case assumed solution provides a safely
18 operable power handling capability for a given driving node, significant
19 disadvantages are that when the design is implemented in hardware, more
20 “real estate” area on the complex logic circuit’s actual integrated circuit(s)

1 and the resulting false need for a larger driving transistor slows down the
2 actual operation of the circuit.

3 Thus, it will be appreciated by those skilled in the art of logical design
4 of complex digital circuits that it would be highly desirable to provide an
5 efficient way to analyze the largest real loading of each and every driving
6 node and to use that information to more correctly report upon violations in
7 the power requirements for the driving transistor and to aid in properly sizing
8 the transistors. This proper sizing may result in a circuit of less area and
9 higher speed.

10 Objects of the Invention

11 It is therefore a broad object of this invention to provide a method for
12 determining the worst case “real” loading of a driving node in a complex
13 logic circuit under “active” or “real” conditions.

14 It is a more specific object of this invention to provide a method which
15 adds logic for mathematical analysis by a formal proof program to analyze
16 the entire complex logic circuit and then uses the results to identify loading
17 violations which exist only under conditions which are logically possible.

18 Summary of the Invention

19 Briefly, these and other objects of the invention are achieved by a
20 process for determining the optimum load driving capacity for each driving

1 node in a complex logic circuit. First, the logic equations of the logic circuit
2 are extracted from an electronic circuit description. Then, the fan-out of each
3 driving node is analyzed to determine if the total number of pass transistor
4 loads of the analyzed node is excessive compared to a predetermined driving
5 capacity. For each flagged driving node, logic equations are added which
6 represent the sum of the driving node's pass transistor loads, and further
7 logic equations are added to compare the number of pass transistors turned
8 on from one to the absolute maximum for the flagged driving node. Then, a
9 formal proof program is used to analyze the logic circuit and determine
10 which of the comparators have a true output. For each flagged node, the
11 comparator for the largest number which has a possible true output is
12 identified to determine the highest possible actual load for the flagged
13 driving node; and, if necessary, the driving capacity of the node is adjusted to
14 handle the determined highest possible actual load.

15 Description of the Drawing

16 The subject matter of the invention is particularly pointed out and
17 distinctly claimed in the concluding portion of the specification. The
18 invention, however, both as to organization and method of operation, may
19 best be understood by reference to the following description taken in

1 conjunction with the subjoined claims and the accompanying drawing of
2 which:

3 FIG. 1 is a logic diagram of a typical driving node of a first exemplary
4 type;

5 FIG. 2 is a logic diagram of another typical driving node in a second
6 exemplary type;

7 FIG. 3 is a process chart illustrating the manner by which the driving
8 capacity of a given driving node has been established according to the prior
9 art; and

10 FIG. 4 is a process chart illustrating the manner by which the
11 requirement for driving capacity of a given driving node is established in
12 accordance with the present invention.

13 Description of the Preferred Embodiment(s)

14 Referring first to FIG. 1, an exemplary node includes a driver 1 and
15 loads 3, 5, 7, 9 selectively imposed on driver 1 according to the input status
16 of pass transistors 2, 4, 6, 8. The pass transistors are individually enabled by
17 suitable signals applied to their respective gates 2G, 4G, 6G, 8G. Similarly,
18 FIG. 2 shows a second exemplary node in which the load 7 may be imposed
19 on the driver 1 if either or both control transistors 6A, 6B are enabled. The
20 driver nodes shown in FIGs. 1 and 2 are representative of circuits widely

1 found in complex logic, and the transistors are typically input stages to
2 certain types of logic elements such as latches, switches, multiplexers, etc.
3 Often, an individual driver will selectively provide switching current to more
4 than four loads as shown, but this is a sufficient number to explain the
5 invention.

6 Early in the design of a complex logic circuit (often a module of a
7 larger circuit system such as a processor), a logic equation generation and
8 analysis software tool is used to extract logic equations from a circuit
9 description. For example, the software tool "Circuit" (from Bull Worldwide
10 Information Systems) will perform this function. The logic equations
11 generated by Circuit and equivalent software tools will identify and configure
12 driver nodes along with other logic and will provide a certain amount of
13 analysis such as determining how many loads are driven by a given driving
14 node. However, the "real" loading is not determinable by such tools, and
15 therefore, the worst case is assumed with the resulting drawbacks discussed
16 above.

17 Thus, attention is directed to FIG. 3 which is a process flow chart
18 according to the prior art. At step 10, the software tool extracts the logic
19 equations for a given logic circuit from a circuit description. At step 11, the
20 software tool (or another suitable software tool) analyzes the fan-out of every

1 driving node to determine the amount of loading driven by each node, some
2 or all of which may be loading “through” pass transistors. At step 12, the
3 software tool marks each node which has potentially too many loads. That is,
4 a standard driver capable of driving a predetermined number of loads may be
5 assumed for all drivers or for each of a class of drivers. Thus, if a given
6 driver node, such as the driver 1 of the nodes shown in FIGs. 1 and 2, has the
7 capability of driving three loads and there are four potential loads as shown,
8 than that given driver node is marked and identified with an error code. At
9 step 13, the drive handling capacity specification for the given node is
10 adjusted higher such that, when the circuit is implemented in hardware, the
11 worst case condition can be handled.

12 Consider now, with reference to FIG. 4, the following discussion of the
13 subject invention which solves the above-discussed problems with the prior
14 art procedure as set forth immediately above with reference to FIG. 3.

15 First, as in the prior art, use a software tool which extracts the logic
16 equations for a given logic circuit from a circuit description, step 20. Then,
17 also as in the prior art, analyze, at step 21, the fan-out of every driving node
18 to determine the total number of pass transistor loads driven by each node.
19 At step 22, also as in the prior art, flag each node which has potentially too

1 many loads (for an assumed “standard” driver which has a predetermined
2 load driving capacity which is less than “worst case”).

3 Step 23 begins a fundamental departure from the prior art. During step
4 23, for those nodes identified as having potentially too many loads, equations
5 are added to the circuit’s logic description to represent the sum of the loads
6 through all transistors that are “turned on”.

7 At step 24, further equations are added to the circuit’s logic description
8 to compare the amount of load seen through the transistors that are “turned
9 on” with each specific number from “one” up to the total (maximum) number
10 of loading transistors. (So if the total number is four as in the example of
11 FIG. 1, there will be four comparisons added to the circuit description, one
12 comparing the number of transistors turned on to “one”, a second comparing
13 the number of transistors turned on to “two”, and then three, and then four).
14 For loads that are calculated as integral units, this comparison can be an
15 equivalence comparison. For loads that are not integral units, the comparison
16 must be an arithmetic comparison such that each comparator compares the
17 sum of all loads with a number representing the drive capacity of the circuits
18 or transistors available as “drivers”.

19 Thus, after steps 23 and 24, the supplemented logic equations for the
20 exemplary node shown in FIG. 1 would be in the general form:

1 [original logic equations for the component including the node]
 2 [a loading sum equation representing the sum of loading
 3 (unit of loading if transistor one is turned on) .plus.
 4 (unit of loading if transistor two is turned on) .plus .
 5 (unit of loading if transistor three is turned on) .plus.
 6 (unit of loading if transistor four is turned on)]
 7 [a loading comparative equation comparing the sum of loading with the
 8 driving capacity of a unit of one,]
 9 [a second loading comparative equation comparing the sum of loading
 10 with the driving capacity of a unit of two,]
 11 [a third loading comparative equation comparing the sum of loading
 12 with the driving capacity of a unit of three,]
 13 [and a fourth comparative equation comparing the sum of loading with
 14 the driving capacity of a unit of four]

15 At step 25, a formal proof program is used, for each identified node, to
 16 analyze the supplemented logic equations and determine which of the loading
 17 comparative equations described in step 24 can have a true output for any
 18 signal input combination (for the entire component including a currently
 19 analyzed node) in which the currently analyzed node can be active. If the
 20 output of the loading comparative equation described in step 24 can never be

1 true, then the condition cannot logically exist and therefore need not be
2 considered with respect to a node's driving capacity requirements. The
3 important consideration is thus which is the largest value of loading which
4 might possibly be seen by the driver, and is represented by largest of the four
5 loading comparative equations might ever be true.

6 For the example, assume that the driver 1 of FIG. 1 is preliminarily
7 established as capable of driving three unit loads simultaneously, but it was
8 found in step 22 that there were potentially four loads on driving gate 1.
9 Without this new methodology, the driving capacity of gate 1 would have to
10 be enlarged to provide for a capacity of four. At step 27 however, the
11 process might determine that the highest load to which the driving node
12 might be subjected is only two so it would be found not to be necessary to
13 increase the driving strength of the driver 1 accordingly (in this example, to
14 full "worst case") and indeed the driving capacity could be reduced to a
15 driver for gate 1 capable of driving only two units of loading.

16 In this manner, the average size of driving nodes in a component can
17 be substantially reduced to achieve the objects set forth above.

18 Thus, while the principles of the invention have now been made clear
19 in an illustrative embodiment, there will be immediately obvious to those
20 skilled in the art many modifications of structure, arrangements, proportions,

- 1 the elements, materials, and components, used in the practice of the invention
- 2 which are particularly adapted for specific environments and operating
- 3 requirements without departing from those principles.